# Exercices: Série 8 - Allocation dynamique - ICC-C 2025-2026

## 1. Échauffement

Écrivez une fonction `inverses` qui prend en paramètre un `size_t n`. Elle doit renvoyer un nouveau tableau de $n$ `double`, de sorte que `resultat[i] == 1.0 / ((double) i)` pour tous les indices `i` du tableau.

Dans `main`, demander une valeur de `n` au clavier. Appelez `inverses`, puis affichez chacun des éléments du tableau reçu en retour.

N'oubliez pas de libérer le tableau quand vous n'en avez plus besoin !

## 2. Revisiter la pile

Reprenez l'exercice sur la pile de la série 6. Vous avez eu besoin de 3 informations pour une pile : le tableau, sa taille maximale, et la taille « actuelle » de la pile. Vous avez dû donner ces 3 (parfois seulement 2) informations à chaque fonction qui voulait manipuler la pile. Ce n'est pas pratique ! C'est pire si on doit manipuler plusieurs piles au sein du même programme : on risque de s'emmeller les pinceaux.

Définissez la structure suivante pour rassembler toutes les informations liées à une pile :

```c
typedef struct stack {
  size_t max_stack_size;
  size_t stack_size;
  int *items;
} stack_t;
```

Définissez les fonctions suivantes pour créer et libérer une pile :

```c
stack_t create_stack(size_t max_stack_size);
void free_stack(stack_t *stack);
```

Réorganisez vos fonctions pour manipuler des `stack_t`, plutôt que de manipuler les 3 informations séparément.

## 3. Tableau dynamique

Nous avons toujours dit qu'une fois un tableau créé avec une taille donnée, il ne peut jamais être redimensionné (changer de taille). Et c'est vrai !

Mais on peut maintenant construire une structure de données qui implémente un « tableau dynamique » de `int`. Définissez une structure `dynarray_t` pour ce faire, ainsi que les fonctions permettant de la manipuler :

- `dynarray_t create_dynarray();` (crée un tableau dynamique avec 0 élément)
- `void free_dynarray(dynarray_t *array);`
- `void dynarray_add(dynarray_t *array, int value);` ajoute un nouvel élément `value` à la fin du tableau dynamique
- `size_t dynarray_size(const dynarray_t *array)` renvoie la taille du tableau
- `int dynarray_get(const dynarray_t *array, size_t index)` renvoie l'élément à l'indice donné.
- `void dynarray_set(dynarray_t *array, size_t index, int new_value)` modifie l'élément à l'indice donné.

- `void dynarray_shrink(dynarray_t *array, size_t new_size)` réduit la taille du tableau en laissant tomber les éléments après `new_size` (qui doit être <= à la taille actuelle).

Contrairement à la pile que nous avons vue dans l'exercice précédent, un tableau dynamique n'a pas de taille maximale. On doit toujours pouvoir l'agrandir.

Écrivez un programme qui utilise vos fonctions pour :

1. Demander des nombres naturels au clavier, un à un (sans demander le nombre total au préalable).
2. S'arrêter quand l'utilisatrice ou utilisateur entre un nombre négatif (ne pas le prendre en compte).
3. Ensuite, *trier* le tableau avec l'algorithme de tri par insertion que vous avez vu en ICC-T (voir semaine 2).
4. Affichez les éléments triés à l'écran.